



Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems

Izosimov, Viacheslav; Pop, Paul; Eles, Petru; Peng, Zebo

Published in:

Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems

Link to article, DOI:

[10.1109/DATE.2005.116](https://doi.org/10.1109/DATE.2005.116)

Publication date:

2005

Document Version

Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):

Izosimov, V., Pop, P., Eles, P., & Peng, Z. (2005). Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems. In *Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems* (pp. 864-869) <https://doi.org/10.1109/DATE.2005.116>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Design Optimization of Time- and Cost-Constrained Fault-Tolerant Distributed Embedded Systems

Viaceslav Izosimov, Paul Pop, Petru Eles, Zebo Peng
Embedded Systems Lab (ESLAB)
Linköping University, Sweden



- Hard real-time applications

- Timing constraints
- Cost constraints

- **Faults**

- Predictable
- **Transient**
- Intermittent

- Hardware solutions

- MARS, TTA, X-by-Wire
 - Permanent faults
 - Costly for transient faults

vs.

- **Software solutions**

- Re-execution/rollback recovery
- Checkpointing/rollback recovery
- Replication, primary-backup...

- Online preemptive

- Flexible

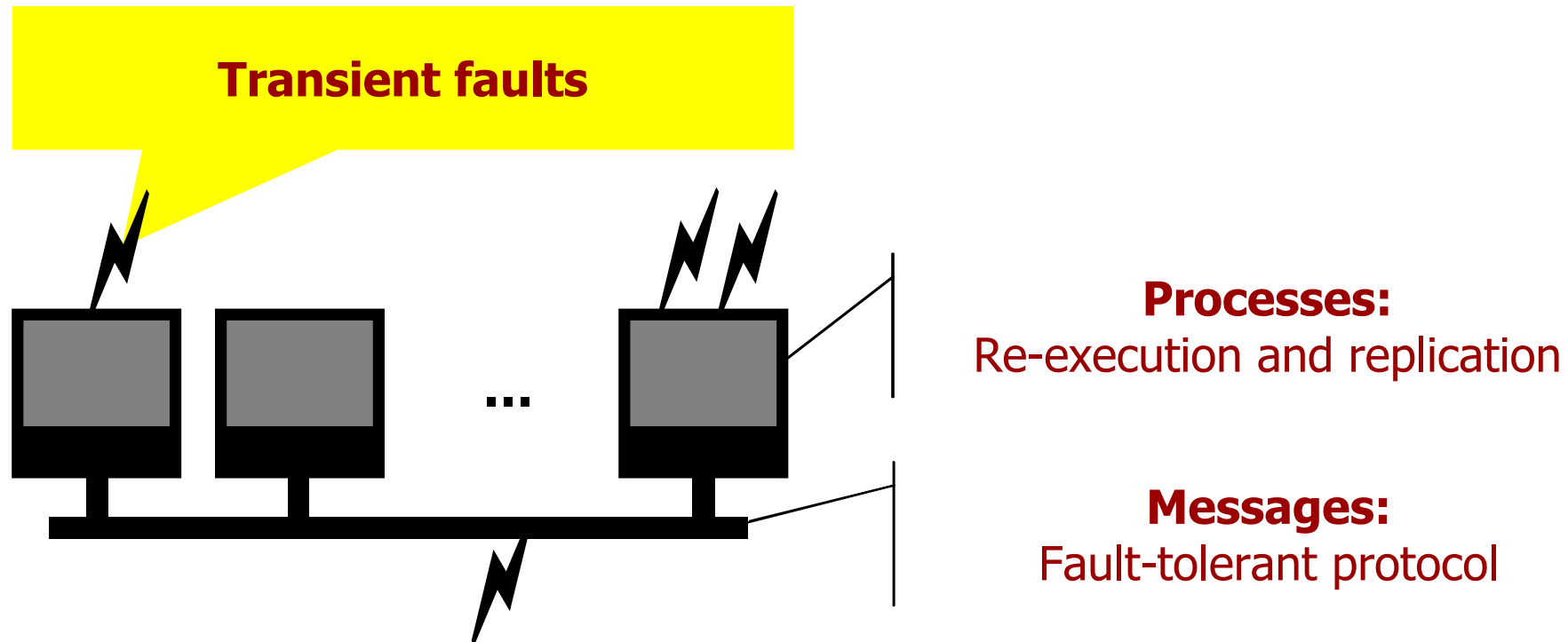
vs.

- **Off-line non-preemptive**

- Predictable

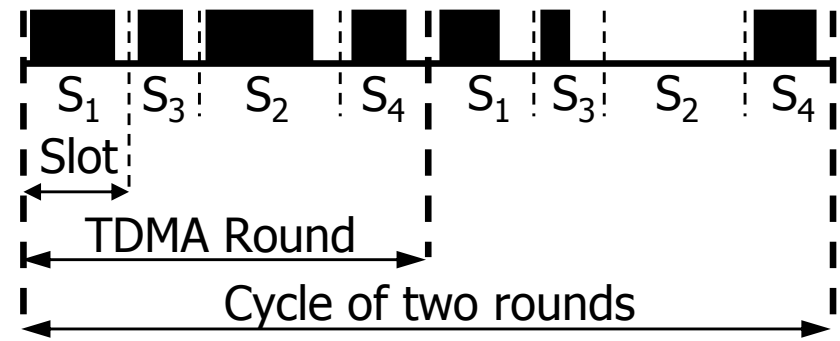
- Motivation
- ➔ System architecture and fault-model
 - Fault-tolerance techniques
- Problem formulation
 - Motivational examples
- Tabu-search optimization strategy
- Experimental results
- Contributions and Message

Fault-Tolerant Time-Triggered Systems



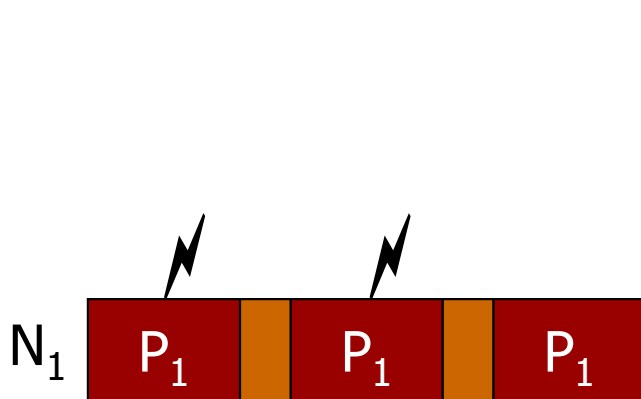
Time Triggered Protocol (**TTP**)

- Bus access scheme: time-division multiple-access (TDMA)
- Schedule table located in each TTP controller: message descriptor list (MEDL)

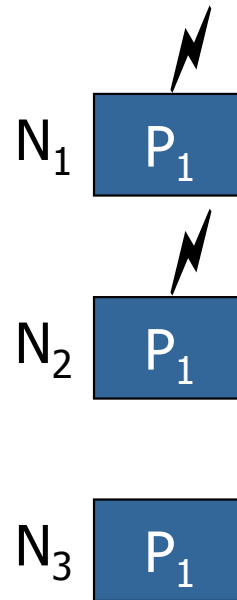




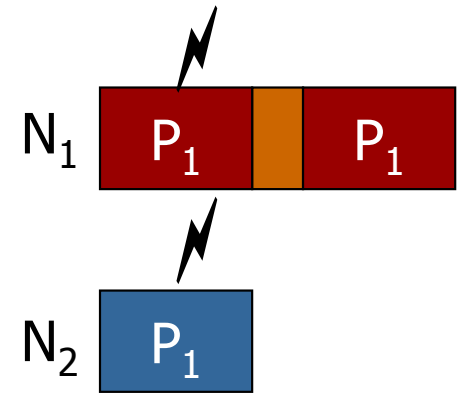
Fault-Tolerant Techniques



Re-execution



Replication



**Re-executed
replicas**

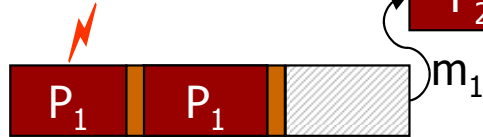
Problem Formulation

- Given
 - **Fault model**
 - Number of transient faults in the system period
 - System architecture
 - Application
 - WCETs, message sizes, periods, deadlines
- Determine
 - **Schedulable and fault-tolerant** design implementation
 - Fault-tolerance policy assignment
 - Mapping of processes and messages
 - Schedule tables for processes and messages

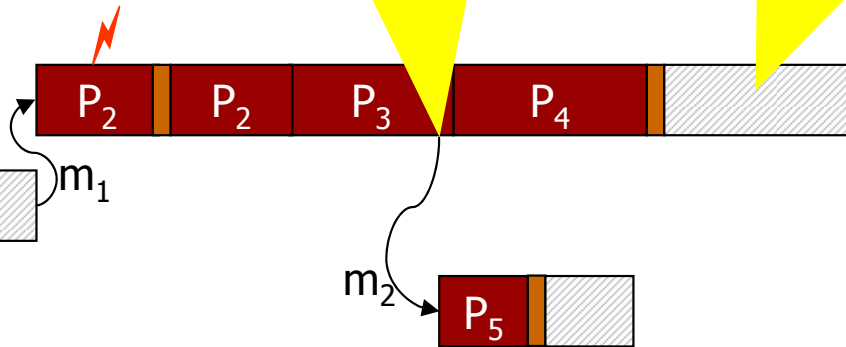
Static Scheduling [Kandasamy et al. 03]

Contingency schedules

$N_1: S_2$
 $N_2: S_{12}$
 $N_3: S_{14}$



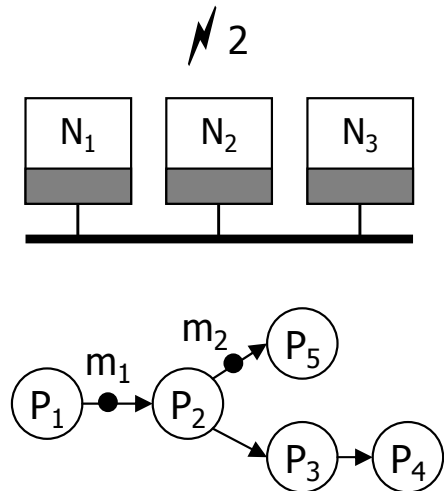
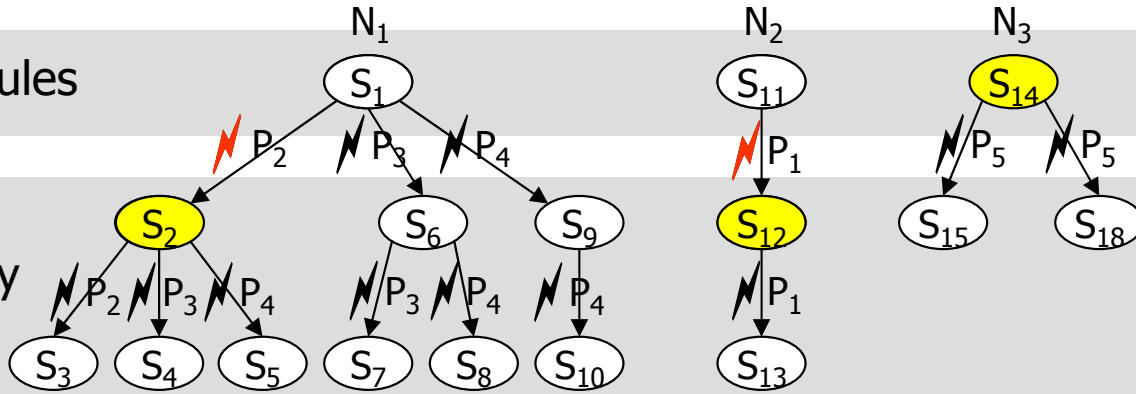
Transparent re-execution



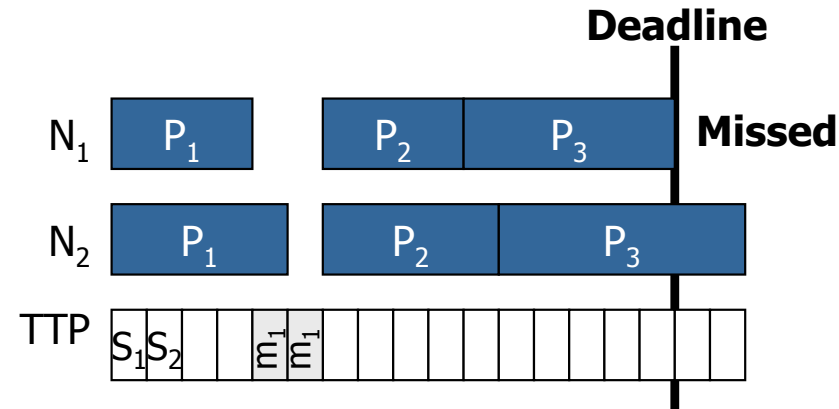
Recovery slack

Root schedules

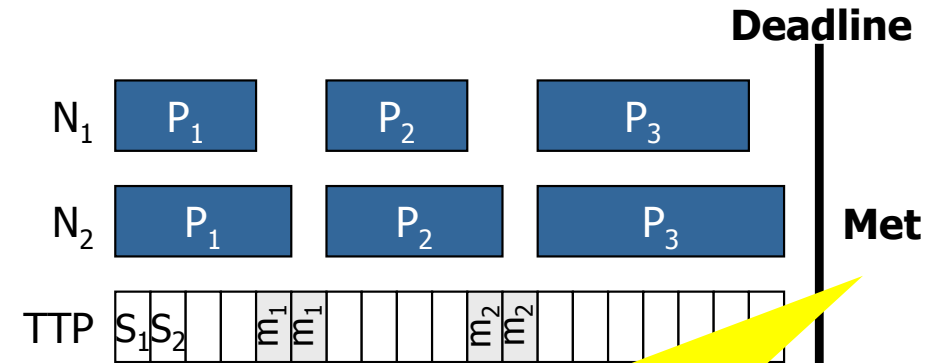
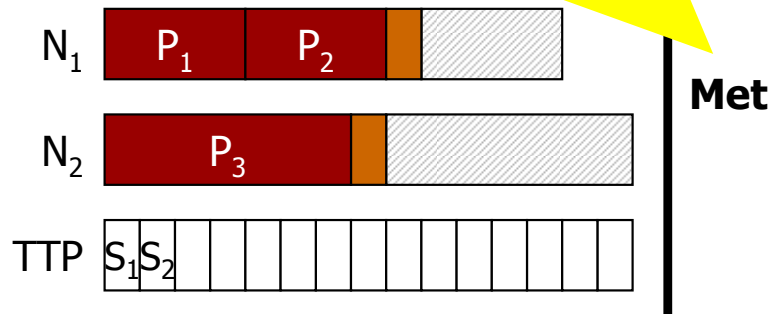
Contingency schedules



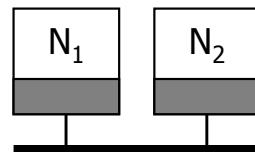
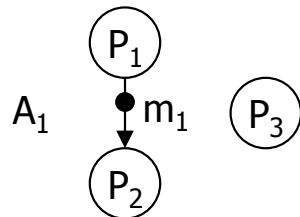
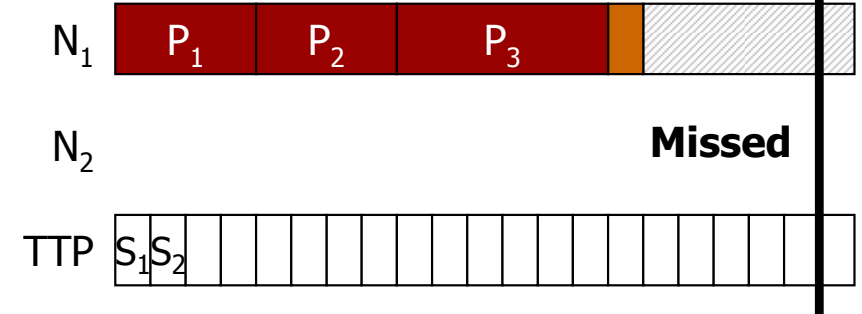
Re-execution vs. Replication



Re-execution is better

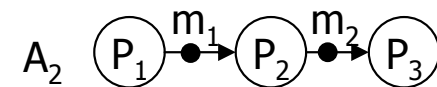


Replication is better

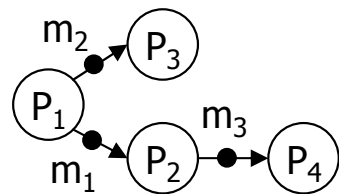
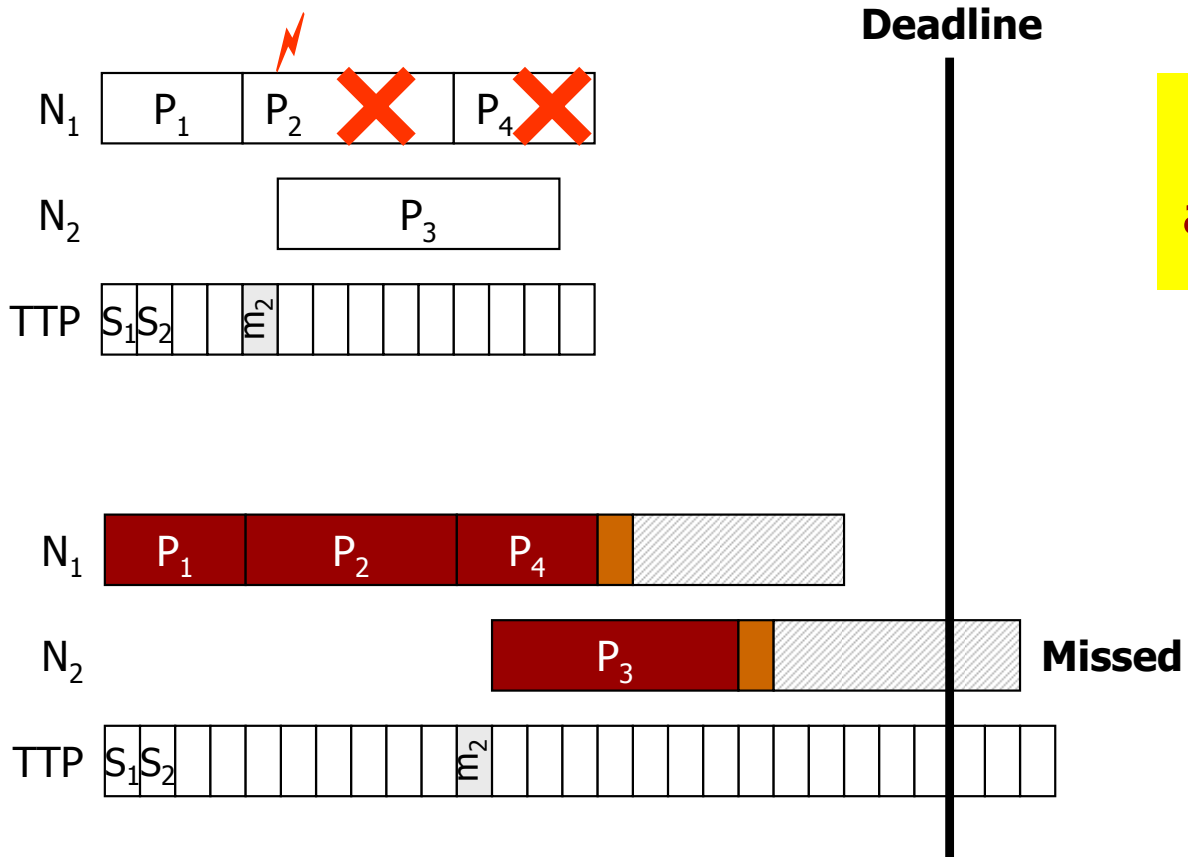


	N ₁	N ₂
P ₁	40	50
P ₂	40	50
P ₃	60	70

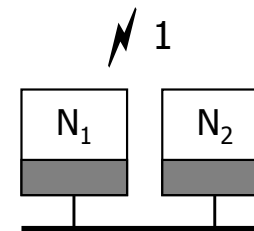
⚡ 1



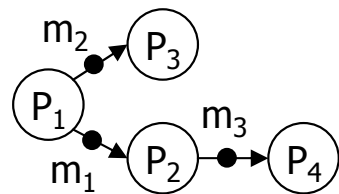
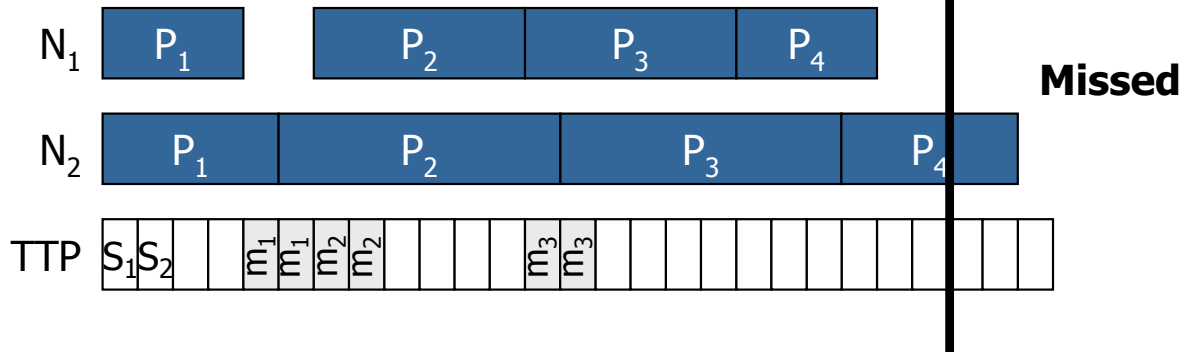
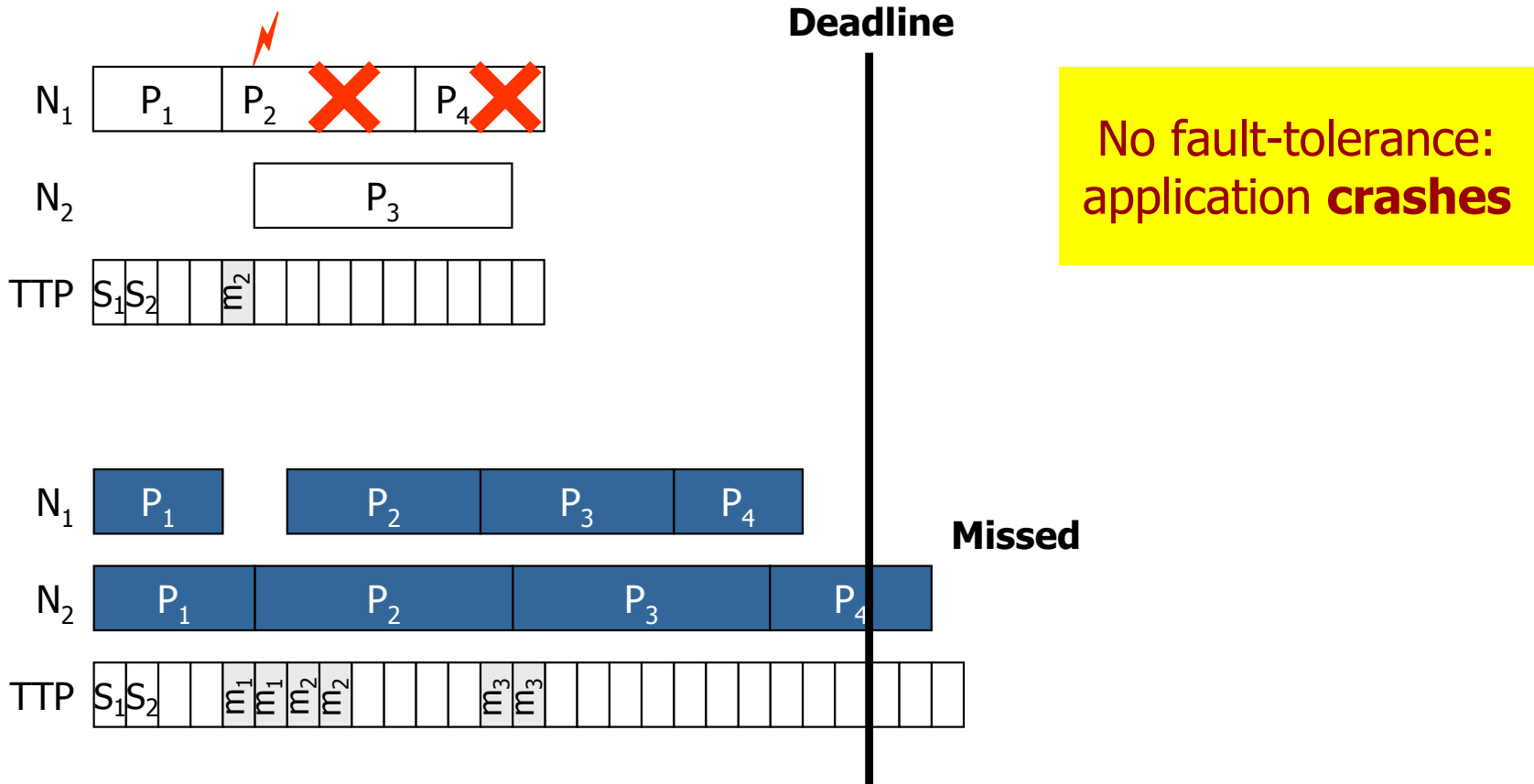
Fault-Tolerant Policy Assignment



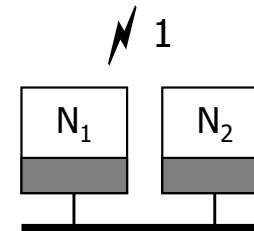
	N_1	N_2
P_1	40	50
P_2	60	80
P_3	60	80
P_4	40	50



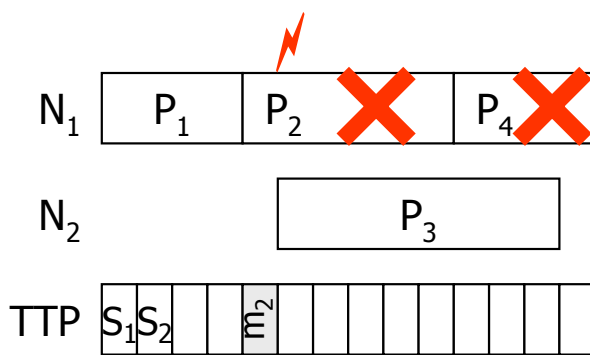
Fault-Tolerant Policy Assignment



	N_1	N_2
P_1	40	50
P_2	60	80
P_3	60	80
P_4	40	50

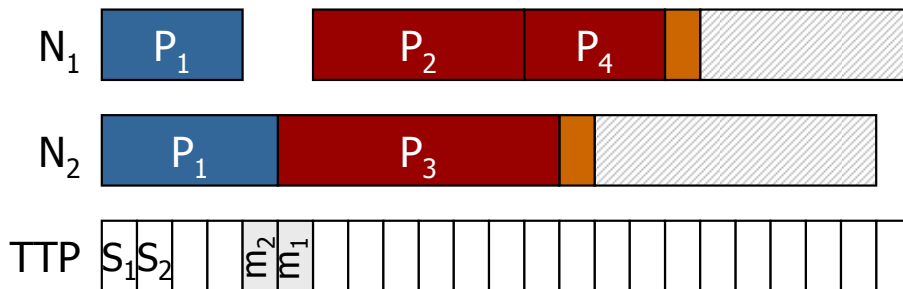


Fault-Tolerant Policy Assignment



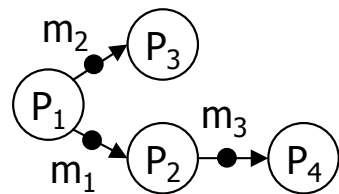
Deadline

No fault-tolerance:
application **crashes**

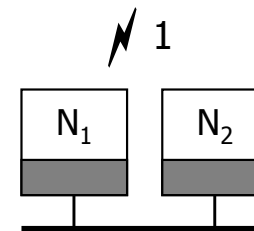


Met

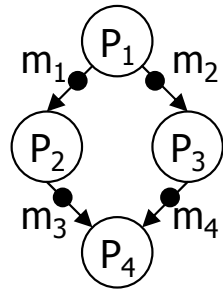
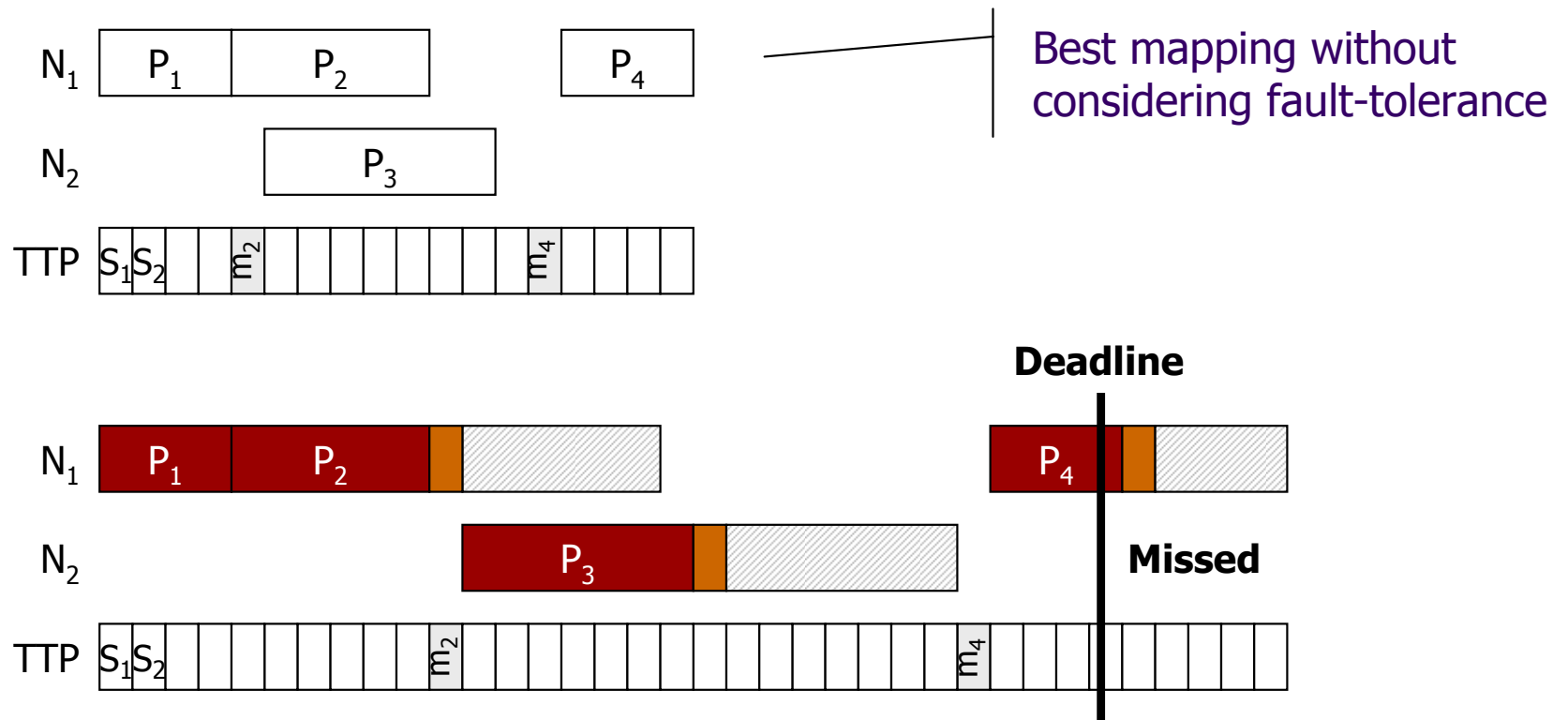
Optimization
of fault-tolerance
policy assignment



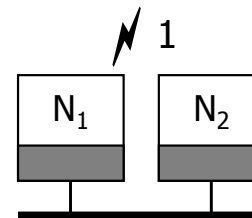
	N_1	N_2
P_1	40	50
P_2	60	80
P_3	60	80
P_4	40	50



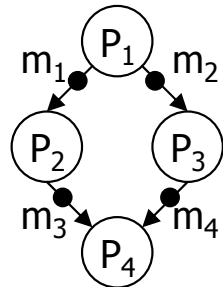
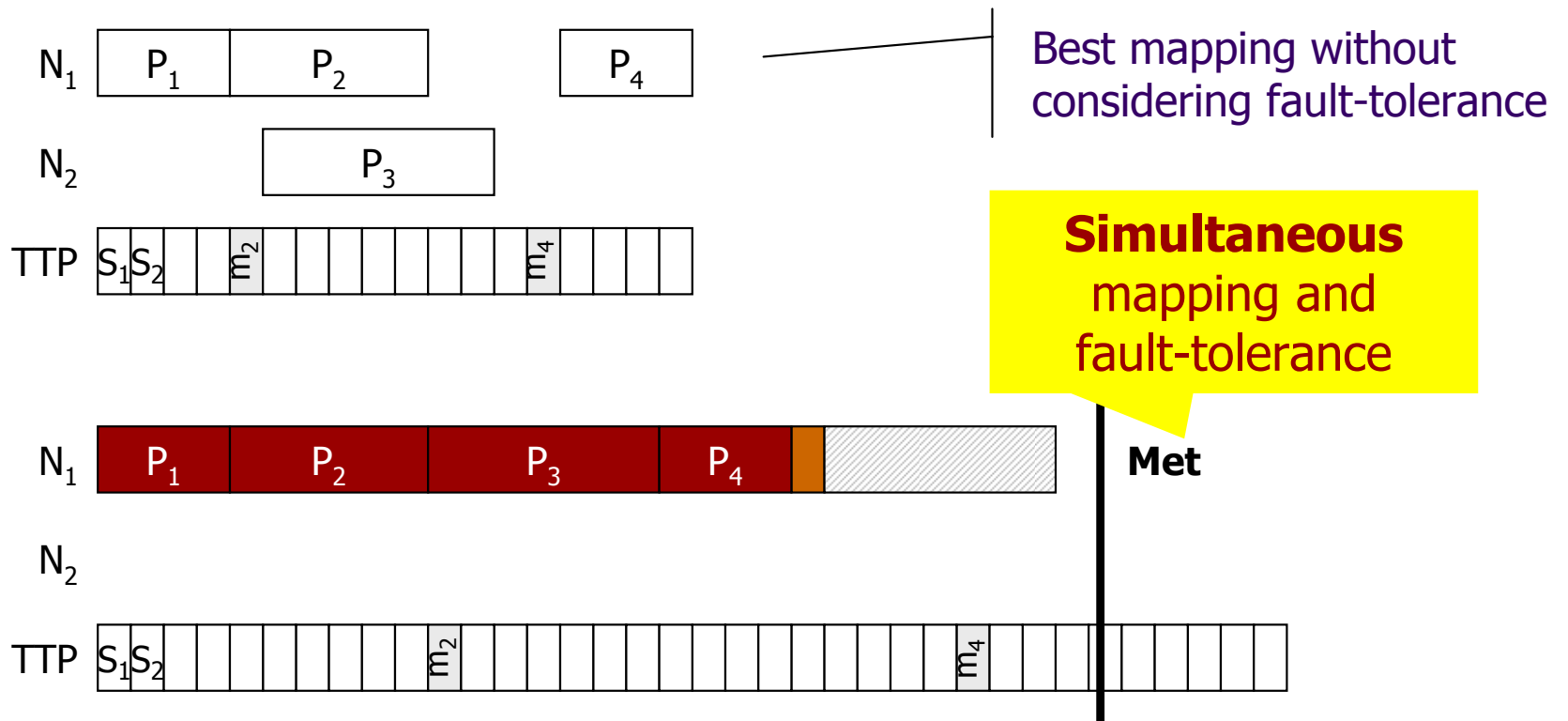
Mapping and Fault-Tolerance



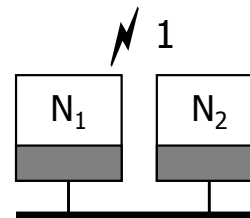
	N_1	N_2
P_1	40	X
P_2	60	70
P_3	60	70
P_4	40	X



Mapping and Fault-Tolerance



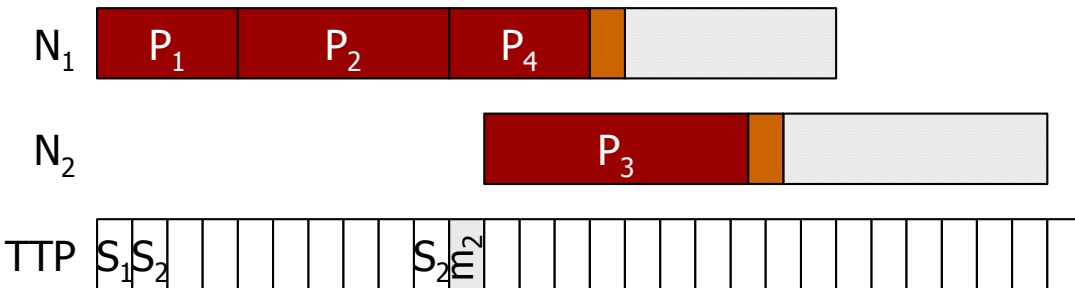
	N ₁	N ₂
P ₁	40	X
P ₂	60	70
P ₃	60	70
P ₄	40	X



Optimization Strategy

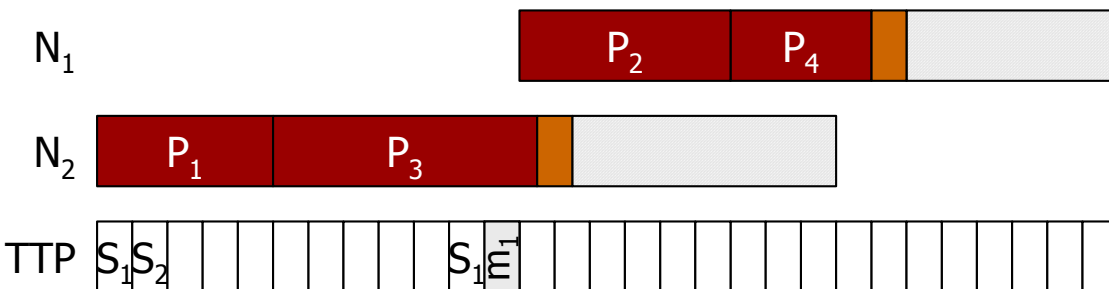
- Design optimization:
 - Fault-tolerance policy assignment
 - Mapping of processes and messages
 - Root schedules
- } **Tabu-search**
- } **List scheduling**
-
- Three tabu-search optimization algorithms:
 1. **Mapping and Fault-Tolerance Policy assignment (MRX)**
 - Re-execution, replication or both
 2. **Mapping and only Re-Execution (MX)**
 3. **Mapping and only Replication (MR)**

MRX Tabu-Search Example



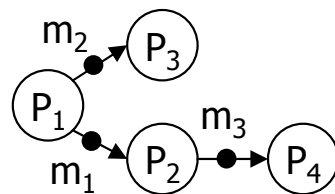
	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

Current solution

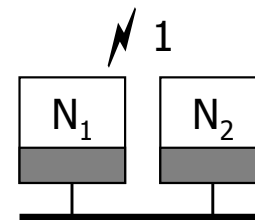


	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

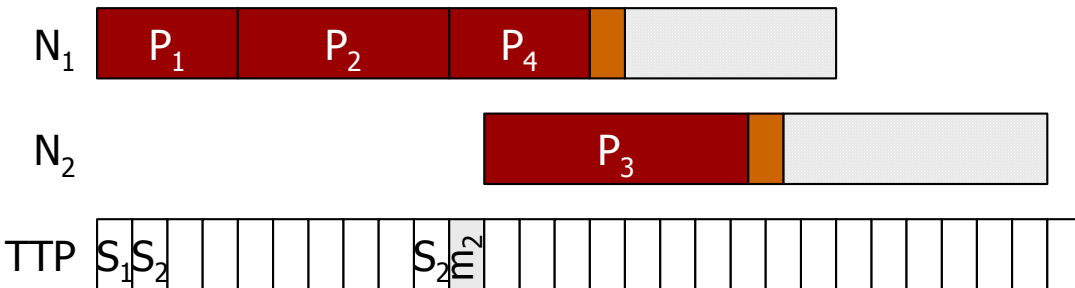
Tabu move & worse than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50

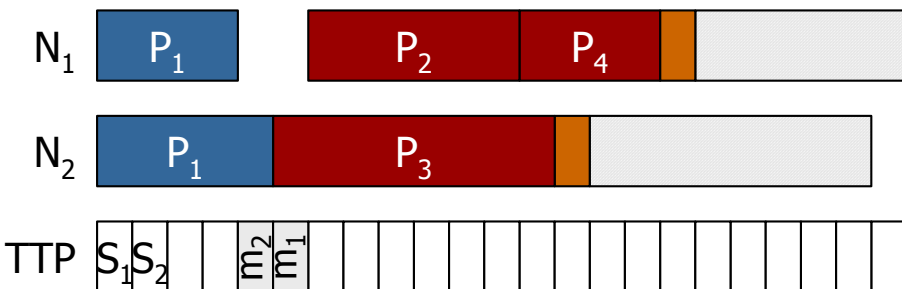


MRX Tabu-Search Example



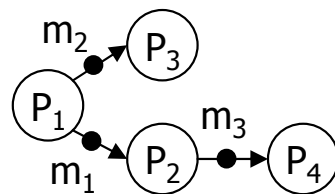
	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

Current solution

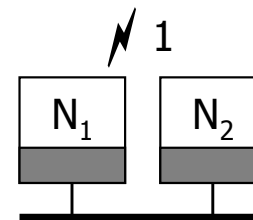


	P_1	P_2	P_3	P_4
Tabu	2	1	0	0
Wait	0	0	2	1

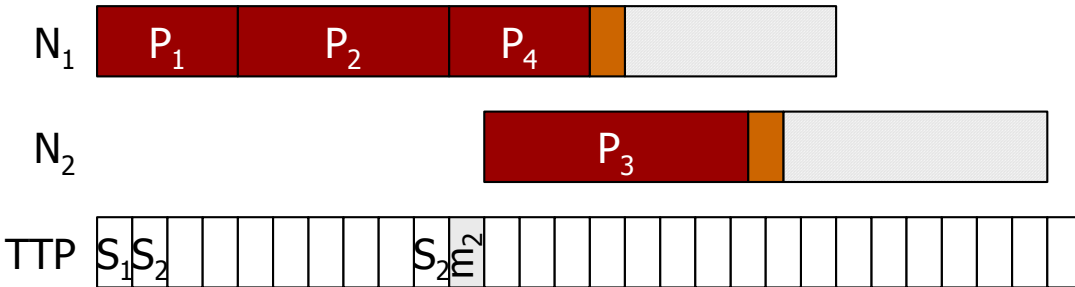
Tabu move & better than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50

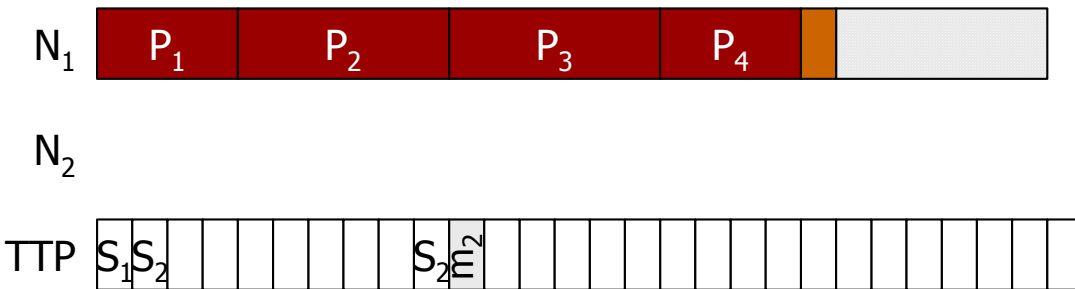


MRX Tabu-Search Example



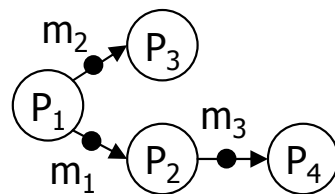
	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

Current solution

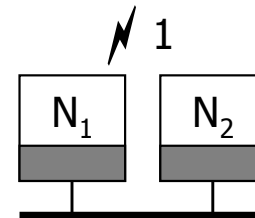


	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

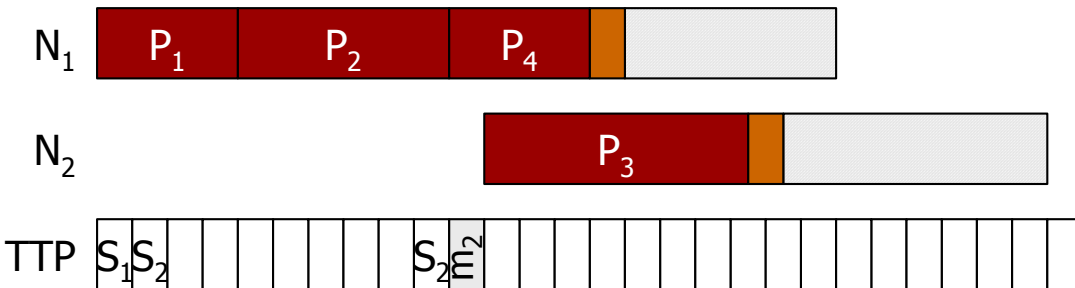
Non-tabu & worse than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50

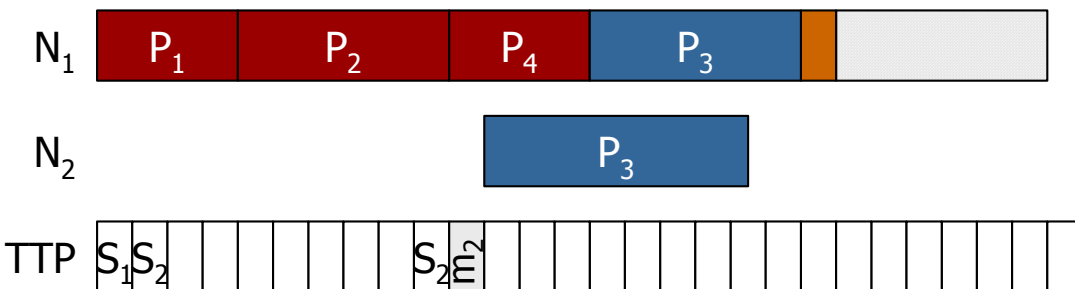


MRX Tabu-Search Example



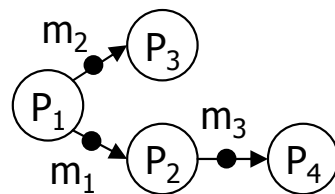
	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

Current solution

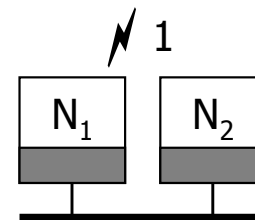


	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

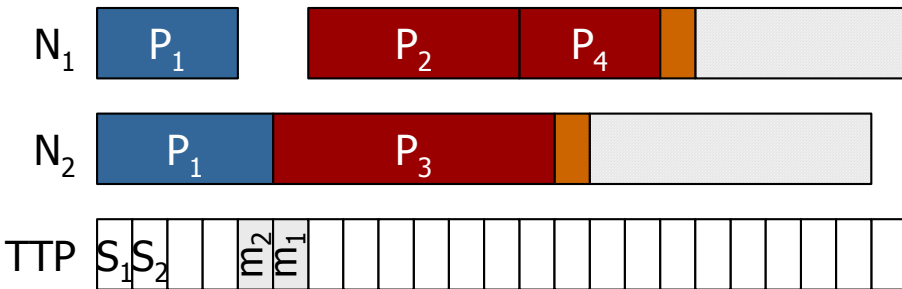
Non-tabu & worse than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50



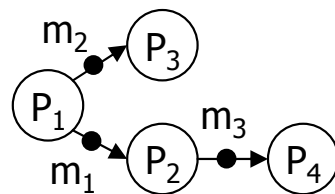
MRX Tabu-Search Example



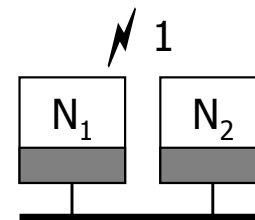
	P_1	P_2	P_3	P_4
Tabu	2	1	0	0
Wait	0	0	2	1

Current solution

Design transformations

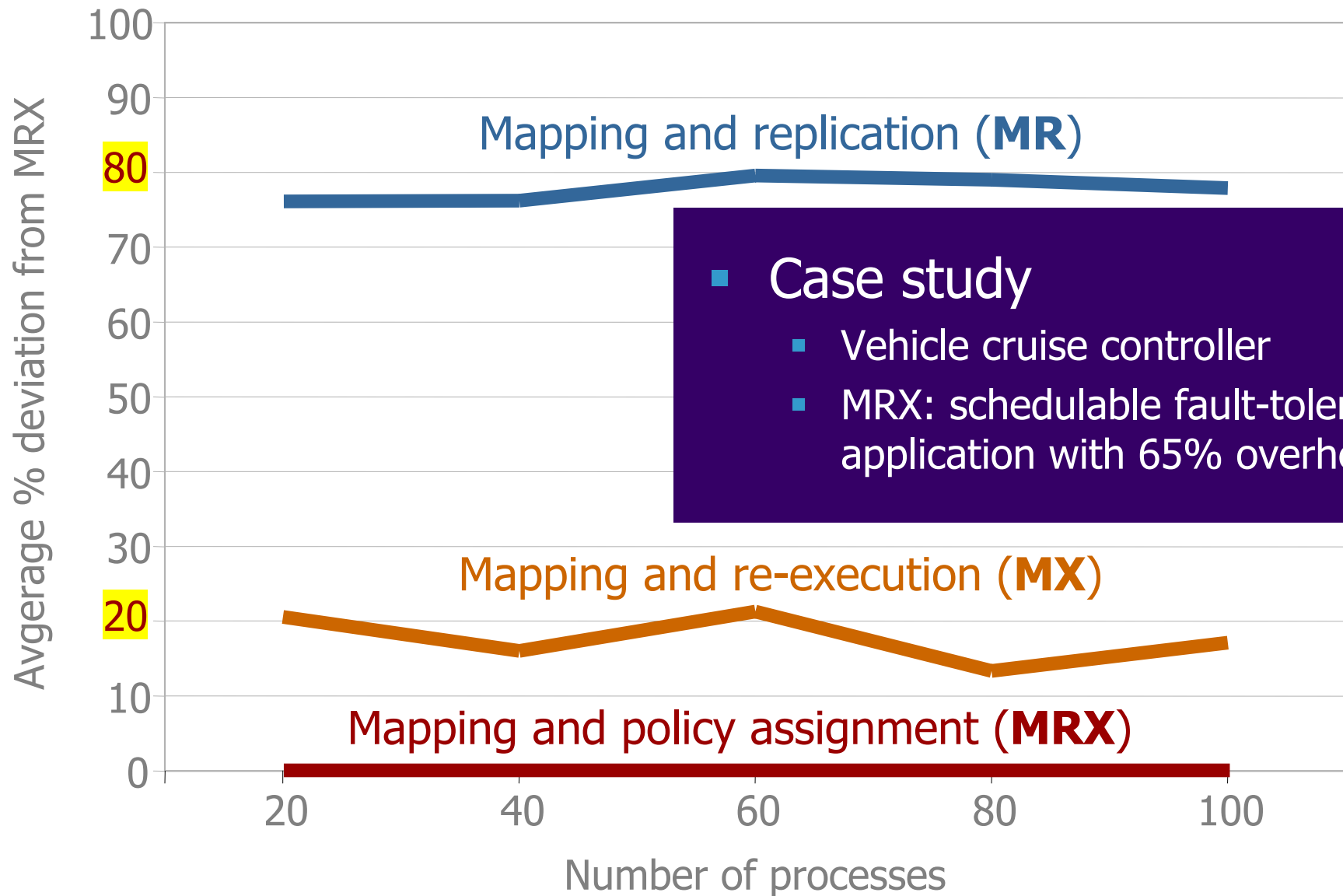


	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50



Experimental Results

Schedulability improvement under resource constraints



Contributions and Message



- Contributions
 - Combined re-execution and replication
 - Optimization algorithms for fault-tolerance policy assignment
 - Efficient contingency schedule generation

Optimization of fault-tolerance
policy assignment needed for
cost-effective fault tolerance